**WE CLAIM:**

1. A computer program product for controlling a computer to detecting an executable computer program containing a computer virus, said computer program product comprising:

analysis logic operable to analyse program instructions forming said executable computer program to identify suspect program instructions being one or more of:

(i) a program instruction generating a result value not used by another portion of said executable computer program; and

(ii) a program instruction dependent upon an uninitialised variable; and

detecting logic operable to detect said executable computer program as containing a computer virus if a number of suspect program instructions identified for said executable computer program exceeds a threshold level.

2. A computer program product as claimed in claim 1, wherein said computer virus is a polymorphic computer virus.

3. A computer program product as claimed in claim 1, wherein said analysis logic is operable to maintain a dependence table indicating dependency between state variables within said computer and loaded variable values.

4. A computer program product as claimed in claim 1, wherein for each program instruction said analysis logic is operable to make a determination as to which state variables are read by that program instruction.

5. A computer program product as claimed in claim 1, wherein for each program instruction said analysis logic is operable to make a determination as to which state variables are written by that program instruction.

6. A computer program product as claimed in claim 3, wherein for each program instruction said analysis logic is operable to make a determination as to which state variables are read and written by that program instruction and for each loaded variable value within said dependence table if any state variable read by that program instruction is marked as dependent upon said loaded variable value, then all state variables written by that program

instruction are marked as dependent upon said loaded variable value with previous dependencies being cleared.

7. A computer program product as claimed in claim 3, wherein said state variables include one or more of:

(i) register values;

(ii) processing result flag values; and

(iii) a flag indicative of a write to a non-register storage location.

8. A computer program product as claimed in claim 1, wherein said analysis logic is operable to maintain an initialisation table indicating which state variables have been initialised.

9. A computer program product as claimed in claim 8, wherein a state variable is marked as initialised upon occurrence of any one of:

(i) a write to said state variable of a determined initialised value; and

(ii) use of said state variable as a memory address value by a program instruction.

10. A computer program product as claimed in claim 1, wherein said analysis logic is operable to parse said executable computer program for suspect program instructions by following execution flow and upon occurrence of a branch first following a first branch path having saved pending analysis results and subsequently returning to follow a second branch path having restored said pending analysis results.

11. A computer program product as claimed in claim 10, wherein a branch path stops being followed when any one of:

(i) there are no further suitable program instruction for execution within that branch path; and

(ii) said branch path rejoins a previously parsed execution path.

12. A computer program product as claimed in claim 1, wherein if said threshold level is exceed, then further virus detection mechanisms are triggered to confirm the presence of a computer virus.

13. A method of detecting an executable computer program containing a computer virus, said method comprising the steps of:

analysing program instructions forming said executable computer program to identify suspect program instructions being one or more of:

(i) a program instruction generating a result value not used by another portion of said executable computer program; and

(ii) a program instruction dependent upon an uninitialised variable; and

detecting said executable computer program as containing a computer virus if a number of suspect program instructions identified for said executable computer program exceeds a threshold level.

14. A method as claimed in claim 13, wherein said computer virus is a polymorphic computer virus.

15. A method as claimed in claim 13, wherein a dependence table is maintained indicating dependency between state variables within said computer and loaded variable values.

16. A method as claimed in claim 13, wherein for each program instruction a determination is made as to which state variables are read by that program instruction.

17. A method as claimed in claim 13, wherein for each program instruction a determination is made as to which state variables are written by that program instruction.

18. A method as claimed in claim 15, wherein for each program instruction a determination is made as to which state variables are read and written by that program instruction and for each loaded variable value within said dependence table if any state variable read by that program instruction is marked as dependent upon said loaded variable value, then all state variables written by that program instruction are marked as dependent upon said loaded variable value with previous dependencies being cleared.

19. A method as claimed in claim 15, wherein said state variables include one or more of:

(i) register values;

(ii) processing result flag values; and

(iii) a flag indicative of a write to a non-register storage location.

20. A method as claimed in claim 13, wherein an initialisation table is maintained indicating which state variables have been initialised.

21. A method as claimed in claim 20, wherein a state variable is marked as initialised upon occurrence of any one of:

(i) a write to said state variable of a determined initialised value; and

(ii) use of said state variable as a memory address value by a program instruction.

22. A method as claimed in claim 13, wherein said executable computer program is parsed for suspect program instructions by following execution flow and upon occurrence of a branch first following a first branch path having saved pending analysis results and subsequently returning to follow a second branch path having restored said pending analysis results.

23. A method as claimed in claim 22, wherein a branch path stops being followed when any one of:

(i) there are no further suitable program instruction for execution within that branch path; and

(ii) said branch path rejoins a previously parsed execution path.

24. A method as claimed in claim 13, wherein if said threshold level is exceed, then further virus detection mechanisms are triggered to confirm the presence of a computer virus.

25. Apparatus for detecting an executable computer program containing a computer virus, said apparatus comprising:

an analyser operable to analyse program instructions forming said executable computer program to identify suspect program instructions being one or more of:

(i) a program instruction generating a result value not used by another portion of said executable computer program; and

(ii) a program instruction dependent upon an uninitialised variable; and a detector operable to detect said executable computer program as containing a computer virus if a number of suspect program instructions identified for said executable computer program exceeds a threshold level.

26. Apparatus as claimed in claim 25, wherein said computer virus is a polymorphic computer virus.

27. Apparatus as claimed in claim 25, wherein said analyser is operable to maintain a dependence table indicating dependency between state variables within said computer and loaded variable values.

28. Apparatus as claimed in claim 25, wherein for each program instruction said analyser is operable to make a determination as to which state variables are read by that program instruction.

29. Apparatus as claimed in claim 25, wherein for each program instruction said analyser is operable to make a determination as to which state variables are written by that program instruction.

30. Apparatus as claimed in claim 27, wherein for each program instruction said analyser is operable to make a determination as to which state variables are read and written by that program instruction and for each loaded variable value within said dependence table if any state variable read by that program instruction is marked as dependent upon said loaded variable value, then all state variables written by that program instruction are marked as dependent upon said loaded variable value with previous dependencies being cleared.

31. Apparatus as claimed in claim 27, wherein said state variables include one or more of:
(i) register values;
(ii) processing result flag values; and
(iii) a flag indicative of a write to a non-register storage location.

32. Apparatus as claimed in claim 25, wherein said analyser is operable to maintain an initialisation table indicating which state variables have been initialised.

33. Apparatus as claimed in claim 32, wherein a state variable is marked as initialised upon occurrence of any one of:

(i) a write to said state variable of a determined initialised value; and

(ii) use of said state variable as a memory address value by a program instruction.

34. Apparatus as claimed in claim 25, wherein said analyser is operable to parse said executable computer program for suspect program instructions by following execution flow and upon occurrence of a branch first following a first branch path having saved pending analysis results and subsequently returning to follow a second branch path having restored said pending analysis results.

35. Apparatus as claimed in claim 34, wherein a branch path stops being followed when any one of:

(i) there are no further suitable program instruction for execution within that branch path; and

(ii) said branch path rejoins a previously parsed execution path.

36. Apparatus as claimed in claim 25, wherein if said threshold level is exceed, then further virus detection mechanisms are triggered to confirm the presence of a computer virus.